**1. Qual seria sua estratégia para identificar as versões? Justifique.**

**Estratégia**: Utilizar o **versionamento semântico** (MAJOR.MINOR.PATCH), pois:

* Fornece clareza sobre a natureza das mudanças;
* É compatível com ferramentas automatizadas;
* Facilita a comunicação entre desenvolvedores e usuários;
* Ajuda a manter compatibilidade e estabilidade ao longo do ciclo de vida do software.

**2. Como nomearia a primeira versão para o público?**

**Resposta**: 1.0.0

**Justificativa**: Segundo o SemVer, a primeira versão estável de um projeto, pronta para uso em produção, deve iniciar com 1.0.0. Antes disso, versões como 0.1.0, 0.2.0 são usadas em desenvolvimento interno .

**3. Após liberado o projeto, uma nova funcionalidade foi requisitada e implementada. Como nomearia esta nova versão?**

**Resposta**: 1.1.0

**Justificativa**: A adição de novas funcionalidades que não quebram a compatibilidade com versões anteriores implica incremento da versão MINOR, mantendo o MAJOR e reiniciando o PATCH em 0.

**4. Considerando a sequência anterior com o esquema de versionamento SemVer. Como ficaria o histórico de versões?**

**Histórico até o momento**:

* 1.0.0 — Primeira versão pública e estável
* 1.1.0 — Adição de funcionalidade sem quebrar compatibilidade

**5. A partir da versão indicada antes, uma série de 3 correções, seguida por duas funcionalidades compatíveis com a versão atual e mais outras 2 correções foram publicadas em série. Qual seria a versão mais recente?**

**Evolução**:

* Correções: 1.1.1, 1.1.2, 1.1.3
* Funcionalidades: 1.2.0, 1.3.0
* Correções: 1.3.1, 1.3.2
* Versão mais recente: 1.3.2

**6. Uma versão nova exigiu mudanças críticas na API, foi lançada e na sequência houveram três novas versões: uma com adição de funcionalidades, seguidas de 2 com correções de bugs. Como fica o histórico?**

**Evolução**:

* 2.0.0 — Mudanças incompatíveis (MAJOR ↑)
* 2.1.0 — Nova funcionalidade (MINOR ↑)
* 2.1.1, 2.1.2 — Correções de bugs (PATCH ↑)

**Histórico**:

* 2.0.0
* 2.1.0
* 2.1.1
* 2.1.2

**7. Pesquise exemplos representativos de versões reais de software considerando cada um dos termos apontados.**

|  |  |  |
| --- | --- | --- |
| Tipo de Versão | Exemplo Real | Descrição |
| Alpha (-alpha) | Python 3.12.0a1 | Primeira fase de testes; instável |
| Beta (-beta) | Ubuntu 24.04 Beta | Prévia para testes públicos |
| RC (-rc) | Node.js 20.0.0-rc.1 | Candidato a lançamento |
| Release Final | Java 17.0.0 | Versão oficial e estável |
| LTS (Long Term Support) | Node.js 18.x LTS, Ubuntu 22.04 LTS | Suporte prolongado e confiável |
| Patch de Correção | React 18.2.0 | Corrige bugs sem adicionar funcionalidades |